

Tema 1 PA

- deadline 3 aprilie 2016 23:55 -

Responsabil Vicentiu Ciorbaru (cvicentiu@gmail.com)

Tema 1 PA

1. Problema 1

1.1 Enunț

1.2 Date intrare

1.3 Date iesire

1.4 Precizari

1.5 Exemplu

2. Problema 2

2.1 Enunț

2.2 Date intrare

2.3 Date iesire

2.4 Precizari

2.5 Exemple

3. Problema 3

3.1 Enunț

3.2 Date intrare

3.3 Date iesire

3.4 Precizari:

3.5 Exemple:

4. Punctare

5. Format arhivă și testare

1. Problema 1

1.1 Enunț

Gigel dorește să programeze primul său joc pe calculator. Neavând prea multă experiență, s-a gândit să pornească cu ceva simplu. Și-a imaginat un joc cu palindroame. Jocul va rearanja literele din cadrul unui palindrom. Scopul jucătorului este să readucă literele amestecate într-o ordine de palindrom. Regulile jocului sunt simple:

- Oricare 2 litere adiacente pot fi interschimbate.
- Jocul se termină când ordinea literelor este corespunzătoare cu cea de palindrom.

Gigel se gândește că trebuie să existe o strategie ce poate rearanja literele într-un număr minim de mutări. Pentru a se asigura că starea inițială a jocului nu este nici prea grea, dar

nici prea usoara, Gigel ar dori să știe în câte mutări ar putea să se termine jocul cel mai repede. El vă roagă să îi testați câteva valori propuse de el.

1.2 Date intrare

Fișierul de intrare **joc.in** va conține mai multe cuvinte a căror litere au fost interschimbate.

Pe prima linie se va afla numărul N de cuvinte.

Pe fiecare din următoarele N linii se va afla un sir de caractere (litere mici ale alfabetului latin).

1.3 Date iesire

Fișierul de iesire **joc.out** va conține N linii. Pe linia i se va afla numărul minim de mutări necesare pentru a transforma șirul de caractere aflat pe linia $i+1$ în fișierul de intrare sau -1 în cazul în care literele nu pot fi rearanjate pentru a se obține un palindrom.

1.4 Precizari

$1 \leq N \leq 1000$

Dimensiune cuvânt ≤ 250

1.5 Exemplu

joc.in	joc.out
4	2
trtaccart	3
iaian	-1
asd	2
aabb	

2. Problema 2

2.1 Enunț

Anii au trecut și Gigel a devenit expert în programarea pe Android. El nu este foarte multumit de rezultatele funcției de recunoaștere vocală de la Google așa că s-a gândit să creeze o soluție proprie.

La rostirea unui cuvânt aplicația îi va returna un vector de variante de lungimi egale, apropiate de cuvântul inițial. El consideră că soluția este suficient de bună dacă **fiecare literă din cuvântul inițial** se găsește în cel puțin una din variante. Pentru a evalua performanța soluției sale Gigel s-a gândit la o funcție:

$$F(\text{cuvant}, \text{variante}) = \text{numărul minim de ajustări asupra cuvântului rostit.}$$

O ajustare se referă la ștergerea unei litere, adăugarea unei litere sau înlocuirea unei litere cu o altă literă în cuvântul inițial.

Având o serie de teste în care caracterele cuvintelor sunt codificate în cod ASCII (fiecare caracter este reprezentat de un număr cuprins între 0 și 255 inclusiv) să se determine ce rezultate vor fi întoarse de funcția de evaluare.

2.2 Date intrare

Fisierul **evaluare.in** va conține pe prima linie 2 valori NR_VAR și N separate prin spațiu, reprezentând numărul de variante, respectiv numărul de caractere din care e compusă fiecare variantă. Pe următoarele NR_VAR linii se găsesc N numere reprezentând codificarea în cod ASCII pentru fiecare variantă. Linia NR_VAR+2 va conține M , numărul de caractere din care e compus cuvântul rostit, iar pe linia NR_VAR+3 se vor găsi M numere reprezentând codificarea cuvântului inițial.

2.3 Date ieseire

Fisierul **evaluare.out** va conține rezultatul întors de funcția de evaluare aplicată pe setul de date din fisierul de intrare, **evaluare.in**.

2.4 Precizari

$NR_VAR \leq 600$

$M, N \leq 1200$

2.5 Exemple

evaluare.in	evaluare.out
3 5 1 5 10 8 7 0 8 9 7 7 1 2 3 4 5 5 1 2 3 4 5	0

Explicatie: Cuvântul rostit este identic cu o varianta, deci nu este nevoie de nicio ajustare.

evaluare.in	evaluare.out
3 5 1 5 10 8 7 0 8 9 7 7 1 2 3 4 5 4 1 11 7 5	2

Explicatie: Cuvântul rostit necesita o inserare pe pozitia 2 a unei valori dintre 5, 8 sau 2 și o modificare a numărului 11 în una din valorile 3, 9 sau 10.

3. Problema 3

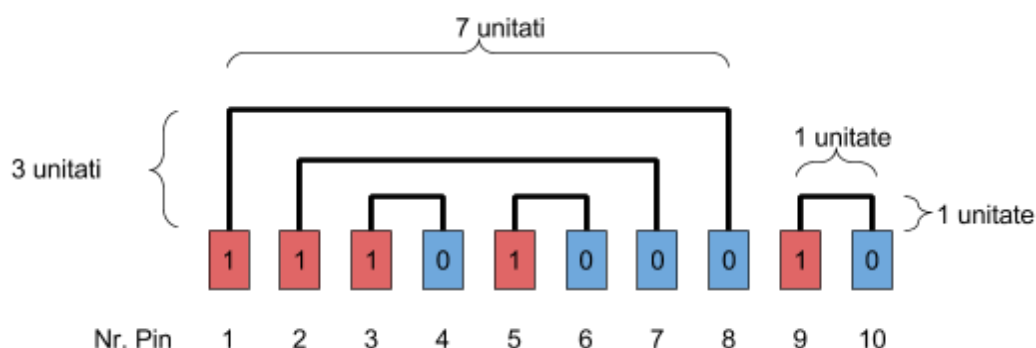
3.1 Enunt

Gigel și-a descoperit o nouă pasiune, **circuitele electronice**. El vrea să își proiecteze propria sa placă de bază. Cum acest lucru este destul de complicat, s-a rezumat la a-și conecta momentan doar un set de pini în configurația cerută de procesor. Gigel are câteva limitări de care trebuie să țină cont:

- Există 2 tipuri de pini, de input (1) respectiv de output (0).
- Toți pinii sunt poziționați pe o singură linie pe placa de bază, la marginea inferioară.
- Un pin de input trebuie să se conecteze neapărat la un singur pin de output. Analog, un pin de output trebuie să se conecteze neapărat la un singur pin de input.
- Traseele de conectare pot fi realizate doar pe verticală și pe orizontală.
- Traseele de conectare nu se pot suprapune.

Întrucât spațiul disponibil este mic, Gigel vă roagă să-l ajutați să traseze cele mai scurte trasee, dându-se configurația pinilor, astfel încât lungimea totală a traseelor să fie minimă.

Un exemplu de cablaj:



În exemplul de mai sus se observă cum fiecare pin de input (1), este conectat la un singur pin de output (0). Traseele nu se intersectează.

Pentru calculul distanței se vor folosi următoarele distanțe:

- Distanța dintre 2 pini adiacenți este de 1 unitate.
- Cel mai mic segment de traseu pe verticală are 1 unitate.

Conform exemplului, distanțele traseelor dintre pini sunt următoarele:

- | | | | |
|---------------------|---|----|----------|
| • Perechea (1, 8): | 3 vertical + 7 orizontal + 3 vertical = | 13 | unități. |
| • Perechea (2, 7): | 2 vertical + 5 orizontal + 2 vertical = | 9 | unități. |
| • Perechea (3, 4): | 1 vertical + 1 orizontal + 1 vertical = | 3 | unități. |
| • Perechea (5, 6): | 1 vertical + 1 orizontal + 1 vertical = | 3 | unități. |
| • Perechea (9, 10): | 1 vertical + 1 orizontal + 1 vertical = | 3 | unități. |
| • Total: | | 31 | unități. |

3.2 Date intrare

Fișierul de intrare **cablaj.in** va conține pe prima linie numărul N al pinilor de tip 1. N reprezintă de asemenea și numărul pinilor de tip 0.

Pe următoarea linie se află o succesiune de cifre 0 și 1, neseparate prin spații, reprezentând configurația pinilor. În total $2 * N$ cifre.

3.3 Date iesire

Fișierul de ieșire **cablaj.out** va conține pe prima linie un număr natural nenul, reprezentând lungimea minimă totală a traseelor de conectare folosite, iar fiecare dintre următoarele N linii va conține două numere naturale x, y , separate printr-un spațiu, reprezentând numărul de ordine a doi pini care se vor conecta; perechile care se vor scrie în fișier vor fi ordonate crescător în funcție de valoarea x .

3.4 Precizari:

$1 \leq N \leq 50$

3.5 Exemple:

cablaj.in	cablaj.out
5 1110100010	31 1 8 2 7 3 4 5 6 9 10

4. Punctare

Punctajul pentru primele două probleme este de 40 de puncte fiecare. A treia problema are un punctaj de 60 de puncte. 10 puncte vor fi acordate pentru coding-style, 10 puncte pentru comentarii și README. Punctajul pe README și comentarii este condiționat de obținerea a unui punctaj pozitiv pe cel puțin un test. Pentru detalii puteți să vă uitați și peste regulile generale de trimitere a temelor.

Prima problemă este obligatorie. Dintre problemele 2 și 3, puteți rezolva una, la alegere. Punctajul maxim care va fi acordat pe tema este de 120 de puncte.

Vor exista mai multe teste pentru fiecare problemă în parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste. Fiecare problemă va avea o limită de timp pe test (precizată mai jos) - dacă execuția programului pe un test al acelei probleme va dura mai mult decât limita de timp veți primi automat 0 puncte pe testul respectiv și execuția va fi întreruptă. În fișierul README va trebui să descrieți soluția pe care ați ales-o pentru fiecare problemă, să precizați complexitatea pentru fiecare și alte lucruri pe care le considerați utile de menționat. Corectorii își rezervă dreptul de a scădea puncte dacă vor considera acest lucru necesar.

5. Format arhivă și testare

Temele pot fi testate automat pe vmchecker - acesta suportă temele rezolvate în C/C++ și Java.

Dacă doriți să realizați tema în alt limbaj trebuie să-i trimiteți un e-mail lui Traian Rebedea (traian.rebedea@cs.pub.ro) în care să îi cereți explicit acest lucru.

Arhiva cu rezolvarea temei trebuie să fie .zip și să conțină:

- Fișierul/fișierele sursă
- Fișierul Makefile
- Fișierul README

Fișierul pentru make trebuie denumit obligatoriu Makefile și trebuie să conțină următoarele reguli:

- build, care va compila sursele și va obține executabilele
- run-p1, care va rula executabilul pentru problema 1
- run-p2, care va rula executabilul pentru problema 2
- run-p3, care va rula executabilul pentru problema 3
- clean, care va șterge executabilele generate

Atentie! Numele regulilor trebuie să fie exact cele de mai sus, în special pentru cele de run. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe testele asociate problemei rezolvate de regula respectivă.

Atenție! Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).

Limitele de timp pentru problema 1 sunt:

- C/C++: X secunde
- Java: X secunde

Limitele de timp pentru problema 2 sunt:

- C/C++: X secunde
- Java: X secunde

Limitele de timp pentru problema 3 sunt:

- C/C++: X secunde
- Java: X secunde